Unified ERC-4337 mempool

[toc]

What does ERC-4337 aim to solve?

ERC-4337 is designed to enable account abstraction without compromising on decentralization and censorship-resistance. It strives to be as decentralized as the block production of the underlying chain.

In practical terms, this means enabling access to smart contract wallets without depending on centralized relayers controlled by a single entity.

It aims to abstract as much of the account properties as possible, including authentication, authorization, replay protection, gas payment, batching & atomicity, etc. All of these aspects are controlled by EVM code rather than rigid concensus rules.

What it means in practice, is that transaction validity must be evaluated through EVM execution, which requires more work and opens up new attack vectors.

What does censorship-resistance require of ERC-4337?

To enable censorship-resistance, the solution needed to satisfy a few requirements, similar to those expected of block-builders:

- Permissionless bundlers that deliver user operations to contracts but cannot change them.
- Users can propagate user-operations to a network where any bundler can serve them.
- Incentivize bundlers to participate in the network and serve user-operations.
- Protect bundlers against griefing by bad actors, including contracts, users, or other bundlers.
- Protect bundlers and the entire network against Denial-of-Service attacks.

These requirements ensure that there will always be non-censoring bundlers willing to serve user operations for a profit, by creating a safe environment where bundlers can operate effectively and profitably.

Even if all the bundlers censor a contract, there is an incentive for anyone to launch a new bundler and capture the profits of this contract's users. The network is permissionless so if there's profit to be made, someone will serve the censored contract for it. In that sense, the network is anti fragile, becoming more robust when facing censorship attacks.

How does ERC-4337 satisfy these requirements?

Certain restrictions are needed to support these requirements. Among them:

- Limit the amount of work a bundler has to do when validating a user operation.
    - Why: bad actors would keep it busy with unpaid work.
    - How: separate validation from execution and limit the gas used during validation.
- Prevent non-deterministic behavior during validation. A validated operation should remain valid as long as the account's state hasn't changed.
    - Why: bad actors could grief bundlers, making them pay for operations and not get compensated.
    - How: ban certain opcodes during validation.
- Prevent mass-invalidation of user operations.
    - Why: bad actors could flood the network with valid operations and then invalidate them cheaply, resulting in too much unpaid work by bundlers, hindering their paid work.
    - How: limit storage access during validation, ensuring that each state change can only invalidate a small number of operations.
- Prevent user operations in a single bundle from affecting each other's validity.
    - Why: bad actors could flood the network with valid but mutually-exclusive operations, and bundlers would perform unpaid work trying to mix and match them unsuccessfully.

    - How: prevent validation from making state changes that affect other validations, and performing all validations before performing any of the user operations.

- Prevent mass-invalidation by entities that need more relaxed storage rules (e.g. certain paymasters, aggregators, and factories).

    - Why: bad actors could deploy a global entity that invalidates a large number of ops by changing its own state, causing a lot of unpaid work and hindering paid work.

    - How: bundlers track a reputation for each entity that takes advantage of the more relaxed storage rules, throttling it down if needed. Reputation required these entities to stake as a method of Sybil-resistance if they need these relaxed rules, so bundlers should verify their stake if they perform an action that requires these rules.

- Prevent entities associated with multiple user operations from causing valid operations to fail in a bundle.

    - Why: a paymaster with a low deposit may be valid for each of the operations, but will run out of funds in the middle of a bundle, not being able to pay for further operations.

    - How: require the bundler to track shared resources while constructing the bundle.

Numerous such scenarios and corner cases were considered by the ERC-4337 team, auditors and other contributors, in order to establish a minimal set of restrictions that retains safety.

The ERC handles these requirements by implementing a complex set of rules that the bundler needs to follow in order to protect itself and the network.

Why must all bundlers implement the same restrictions?

A single centralized bundler could defend itself by throttling RPC calls that cause problems. It could throttle by IP, API key, or any other common DoS protection.

However, such a bundler would be centralized so it might censor its users, or it might be taken down by network-level DDoS or by legal action. Bundlers benefit from being part of a large permissionless network to prevent liveness issues and censorship. Attacking any particular bundler cannot take down the network, so each bundler is less likely to be attacked in the first place. Safety in numbers.

In a permissionless p2p network, common DoS protections cannot be applied the same way. A bad actor could fake any number of peers from any number of IP addresses and perform attacks against legitimate bundlers in the network.

DoS protection in p2p networks relies on dropping and banning bad peers. If a bundler receives invalid UserOps from a peer that claims they are valid at the current state, or if it receives UserOps that should have been blocked by the rules, it drops and blacklists that peer.

Therefore, if different bundlers don't implement exactly the same policy, anyone could flood the network with UserOps that have different validity results in different bundlers. This would cause the bundlers to blacklist each other, resulting in a fragmented mempool. A large number of small mempools.

Why is this bad?

- Each of these pools is smaller, more centralized and prone to censorship and attacks.

- Each of them probably uses a single implementation across all pool members, being more prone to exploitation of software bugs. The bundlers ecosystem will have implementations in different languages, similarly to Ethereum clients, resulting in a robust network. A mempool fragmented by different rules will lose that property.

- Each of them is less profitable due to only seeing a small part of the entire global mempool. Less profit means less incentive to run bundlers, leading to a shrinking and less robust network.

How can we prevent this mempool fragmentation?

Multiple implementations make the network robust, but only if they are compatible.

Ethereum faced the same challenge, having multiple clients implemented using different programming languages and software methodologies. They all implement the yellow paper, but any incompatibility could lead to Ethereum forking to multiple chains.

To ensure compatibility, Ethereum uses reference implementations such aspy-evm, a test suite and fuzzing frameworks meant to identify corner cases where compatibility might break

ERC-4337 can be thought of, as the yellow paper of the bundlers network. But a paper is not enough. We need a reference implementation and a test suite that any bundler can integrate to ensure compatibility with the mempool.

The ERC-4337 team has been building this bundlers test suite. It is still work-in-progress and we invite all bundler implementers to contribute missing tests until we reach full coverage of the ERC-4337 rules.

The initial tests cover the rules to ensure compatibility and safety, but can't test actual integration because the p2p specs are TBD. As soon as the p2p layer is defined, we will also add tests to ensure network-level compatibility.

There are actually two types of tests:

- Compatibility tests. Tests for conditions that would result in mempool fragmentation.
    - For example, a bundler that incorrectly impelements opcode banning and accepts a UserOp which uses `block.number` during validation, it should fail compatibility tests.
    - These tests should be seen as mandatory for participating in the mempool.
    - There is a finite number of these tests, as they only need to cover the validity rules.
- Safety tests. Tests for conditions that wouldn't lead to peer banning and fragmentation, but may indicate vulnerability to DoS attacks against that bundler.
    - For example, a bundler that incorrectly implements the reputation system will not cause fragmentation because reputation is local to each bundler, but it will have a hard time mitigating mass-invalidation attacks by rogue paymasters.
    - These tests are optional, but a bundler that chooses to skip such a test should find other ways to mitigate the risk demonstrated by that test.
    - Tests in this category will be added whenever someone comes up with an attack vector missed by existing tests.

How can ERC-4337 enable safe exceptions to the rules?

"*But wait! There's this one contract that I know to be safe. I want to whitelist it*".

While the ERC makes every effort to ensure bundler safety by applying restrictions, it is possible that a contract will be audited specifically to ensure that it cannot be used to facilitate an attack on bundlers (e.g. cannot cause mass invalidation or non-deterministic validation) even though it violates certain restrictions. It shouldn't be common, as most use cases can be supported without breaking the rules, but some may require it.

The canonical mempool cannot make assumptions about contracts, but a group of bundlers can. The ERC supports this though alternative mempools.

A group of bundlers can decide to support an exception to the rules, such as whitelisting a known contract, by forming another mempool with a different ID. These bundlers can still participate in the canonical mempool (where they must apply the ERC rules without exceptions) while also participating on the alternative pools. UserOps that conform to the ERC rules will be propagated to the canonical mempool, and ones that don't - but that conform to the alternative mempool's rules, will be propagated only to the participants of that mempool.

The bundler is incentivized to still participate in the canonical mempool even if it is a member of another one, because it strictly improves its profitability by selecting the most profitable UserOps across both mempools. The mempool rules are enforced by bundlers but not by the EntryPoint contract, so a bundler can mix UserOps of multiple mempools in the same bundle, building the most profitable bundle.

A bundler may participate in any number of such mempools, but it should only join ones where the exceptions have been audited for safety and cannot be used to attack it. The ERC only ensures the safety of the canonical mempool. Any other mempool should be evaluated separately by its participants.